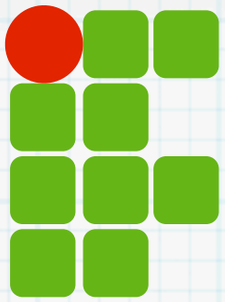
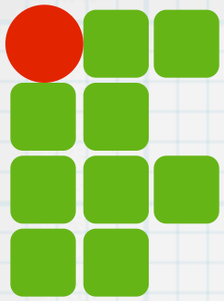


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Mais arrays

Jorgiano: jorgiano.vidal@ifrn.edu.br
Ivanilson Júnior: ivanilson.junior@ifrn.edu.br

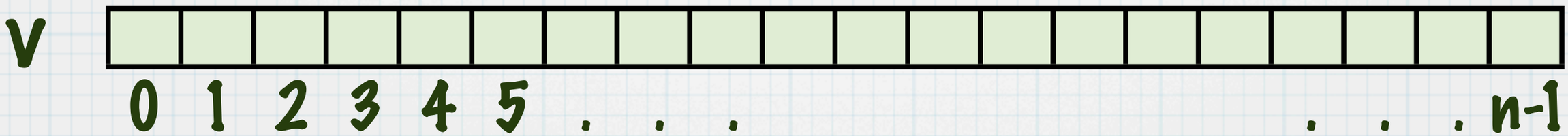


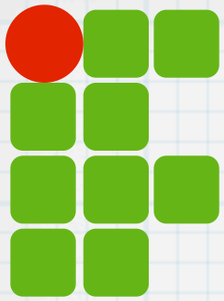
O que veremos hoje?

* Métodos de arrays
  * uniq
  * map
  * select
  * inject

Arrays

- Coleção de elementos
  - Comum realizar operação sobre todos os elementos a coleção





uniq

* Elimina elementos duplicados

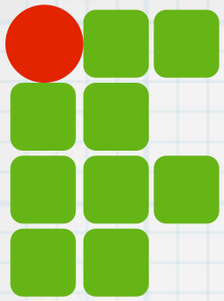
```
x = [1,2,3,4,3,1,2,3,1,2]
y = x.uniq
```

x

1	2	3	4	3	1	2	3	1	2

y

1	2	3	4



map

* Processa um bloco de código para cada elemento do array e gera um array com os resultados do processamento

```
x = [1,2,3,4]
quad = x.map do |n|
   n*n
end
```

Para cada elemento de x retorna o seu quadrado

x	1	2	3	4

quad	1*1	2*2	3*3	4*4

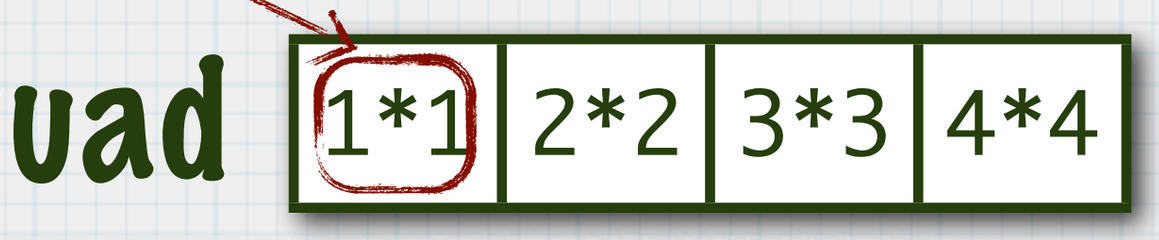
map

* Processa um bloco de código para cada elemento do array e gera um array com os resultados do processamento

```
x = [1,2,3,4]
quad = x.map do |n|
  n*n
end
```

parâmetro para cada processamento

Para cada elemento de x retorna o seu quadrado



Blocos

* Permite processar trecho de código recebendo parâmetros
* Vários métodos de arrays usam blocos
* Torna o código mais sucinto

```
x = [1,2,3,4]
x.método do |p|
   BLOCO
end
```

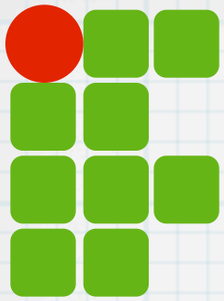
BLOCO é processado uma vez para cada elemento do array, cujo valor é atribuído a p

map

* Criar um novo array com dados lido do teclado

```
x = [0,1,2,3,4]
y = x.map do gets.to_i end
```

```
y = 5.times.map do gets.to_i end
```



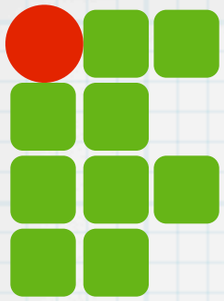
map

* Criar um novo array com dados lido do teclado

```
x = [0,1,2,3,4]
y = x.map do gets.to_i end
```

```
y = 5.times.map do gets.to_i end
```

Processe o bloco 5 vezes



select

* Realiza um filtro no array
  * Gera um array com os elementos que satisfaçam um critério de seleção

```
x = []
for i in 1..100 do
  x << i
end
y = x.select do |n|
  n%3==0
end
```

novo array com todos os elementos de x múltiplos de 3

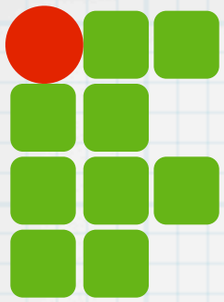
select

* Realiza um filtro no array
  * Gera um array com os elementos que satisfaçam um critério de seleção

```
x = []
for i in 1..100 do
  x << i
end
y = x.select do |n|
  n%3==0
end
```

novo array com todos os elementos de x múltiplos de 3

```
y = (1..100).select do |n|
  n%3==0
end
```



select

* Números pares

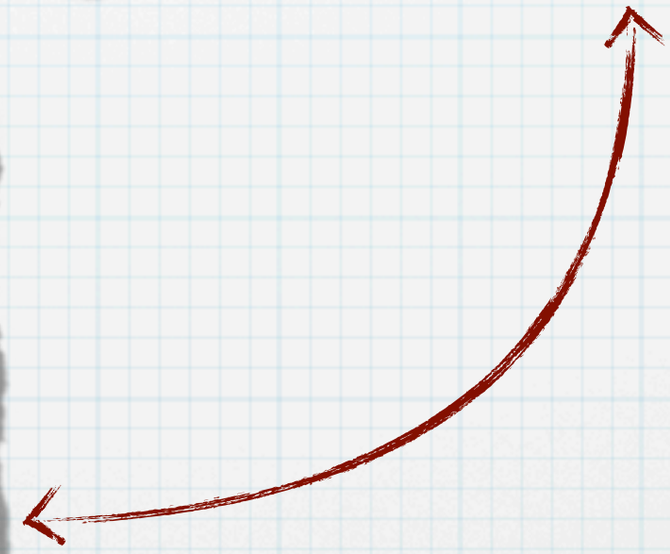
```
y=[]
for n in x do
  if (n%2==0) then
    y << n
  end
end
```

select

* Números pares

```
y = x.select do |n|
   n%2==0
end
```

```
y=[]
for n in x do
   if (n%2==0) then
      y << n
   end
end
```



inject

* Aplica operação binária com elementos consecutivos do array
* Usa resultado anterior

```
a = [12,3,13,34,65]
total=a.inject do |soma,num|
    soma + num
end
```

```
[ 12 , 3 , 13 , 34 , 65 ]
```

inject

* Aplica operação binária com elementos consecutivos do array
* Usa resultado anterior

Resultado anterior

elemento

```
a = [12,3,13,34,65]
total=a.inject do |soma, num|
    soma + num
end
```

[12 , 3 , 13 , 34 , 65]

inject

* Aplica operação binária com elementos consecutivos do array
* Usa resultado anterior

Resultado anterior

elemento

```
a = [12,3,13,34,65]
total=a.inject do |soma, num|
    soma + num
end
```

[12 + 3 + 13 + 34 + 65]

inject

* Aplica operação binária com elementos consecutivos do array
* Usa resultado anterior

Resultado anterior

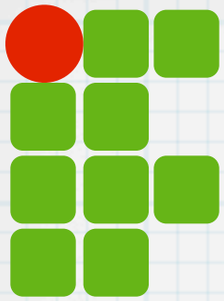
elemento

Valor inicial da soma

```
a = [12,3,13,34,65]
total=a.inject do |soma,num|
    soma + num
end
```

```
a = [12,3,13,34,65]
total=a.inject(0) do |soma,num|
    soma + num
end
```

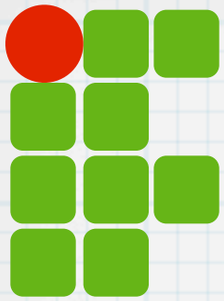
[12 + 3 + 13 + 34 + 65]



inject

* Pode ser simplificado

```
a = [12,3,13,34,65]
total=a.inject do |soma,num|
    soma + num
end
```



inject

* Pode ser simplificado

```
a = [12,3,13,34,65]
total=a.inject do |soma,num|
    soma + num
end
```

mesmo que

```
a = [12,3,13,34,65]
total = a.inject(:+)
```

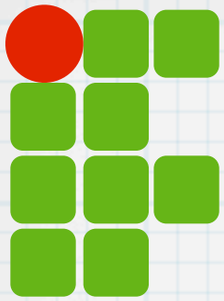
inject

* Pode ser simplificado

```
a = [12,3,13,34,65]
total=a.inject do |soma,num|
    soma + num
end
```

mesmo que

```
a = [12,3,13,34,65]
total = a.inject(:+)
```



inject

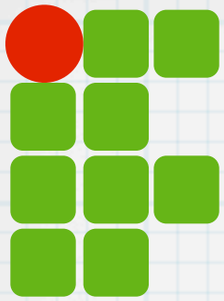
* Multiplicar números de um array

```
a.inject(1) do |a,b|
  a*b
end
```

* Informar se todos os números do array são positivos

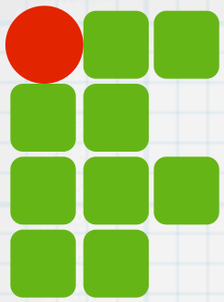
```
a.inject(true) do |a,b|
  a and b>0
end
```

* Achar maior número do array
  * Exercício



Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares



Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

```
resp1=a.map do |n|
  n*n
end
resp2=resp1.select do |n|
  n%2==1
end
resp3=resp2inject(0) do |a,b|
  a+b
end
```

Combinando métodos

* Aumenta o poder de simplificação
  * Exemplo: Dado um array contento 10 números, calcular o quadrado de cada elemento e depois a soma dos quadrados que são números ímpares

map

select

inject

```
resp1=a.map do |n|
  n*n
end
resp2=resp1.select do |n|
  n%2==1
end
resp3=resp2inject(0) do |a,b|
  a+b
end
```

```
resp = a.map do |n|
  n*n
end.select do |n|
  n%2==1
end.inject(0) do |a,b|
  a+b
end
```

Considerações

* O uso de blocos torna o código mais sucinto
* Encoraja o estilo declarativo
* Evitar reatribuição de uma valor a uma variável
  * Mais fácil entender o código
  * Facilita correção de erros
* Documentação completa de arrays em ruby
  * http://www.ruby-doc.org/core-1.9.3/Array.html
    * Acessada em 26/06/2012

Dúvidas?